AFRL-SN-RS-TR-2001-146 Vol IV (of VI)
Final Technical Report
July 2001

KNOWLEDGE BASE APPLICATIONS TO ADAPTIVE SPACE-TIME PROCESSING, VOLUME IV: KNOWLEDGE-BASED TRACKING

ITT Systems

Technology Service Corporation

Charles Morgan and Lee Moyer

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-SN-RS-TR-2001-146 Vol IV (of VI) has been reviewed and is approved for publication.

APPROVED: Michael C Wicks

MICHAEL C. WICKS
Project Engineer

FOR THE DIRECTOR: Robert G. Polce

ROBERT G. POLCE, Chief
Rome Operations Office
Sensors Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/SNRT, 26 Electronic Pky, Rome, NY 13441-4514. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JULY 2001	Final Aug 95 - Jul 99

4. TITLE AND SUBTITLE
KNOWLEDGE BASE APPLICATIONS TO ADAPTIVE SPACE-TIME PROCESSING, VOLUME IV: KNOWLEDGE-BASED TRACKING

5. FUNDING NUMBERS
C - F30602-95-C-0041
PE - 62702F
PR - 4506
TA - 11
WU - 2B

6. AUTHOR(S)
Charles Morgan and Lee Moyer

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Prime: ITT Systems
775 Daedalian Drive
Rome New York 13441

Sub: Technology Service Corporation
6515 Main Street
Trumbull CT 06611

8. PERFORMING ORGANIZATION REPORT NUMBER
TSC-CT109-22

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/SNRT
26 Electronic Pky
Rome New York 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-SN-RS-TR-2001-146
Vol IV (of VI)

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: Michael C. Wicks/SNRT/(315) 330-2556

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This report describes a knowledge-based tracking capability that will support a space-time adaptive processing (STAP) environment. The effort described in this report resulted in development of a multiple target tracker and the design and testing of several knowledge-based rules.

Three types of tracking filters are described:

1. An uncoupled two-state alpha-beta filter with position and velocity component states,
2. An uncoupled three-state Kalman filter with position, velocity, and acceleration component states, and
3. An extended four-state Kalman filter with both x and y position and velocity component states.

The first two filters use a one-dimensional measurement vector containing the report position component. The extended Kalman filters uses a three-dimensional measurement vector containing x and y report position and "pseudo" Doppler, the latter defined as range times range rate.

14. SUBJECT TERMS
Radar, Algorithms, Tracking, Knowledge Base Tracking

15. NUMBER OF PAGES
84

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Table of Contents con't

Table of Contents con't

Table of Figures

1.0 Introduction and Overview

The purpose of this effort has been to provide ITT Systems & Sciences with a knowledge based tracking capability that will support a space time adaptive processing (STAP) environment. This has included the development of a basic multiple target tracker and the design and testing of several knowledge based rules. A rule book containing 25 potential knowledge based rules was developed and is presented in Volume V.

For the purpose of ITT's application, the main elements of the tracker software can be imbedded in a larger STAP simulation by removing the GUI. The key tracking modules are setup_Tracker_6 and Run_Tracker_8.

The Run_Tracker_module allows the use of three types of tracking filters. These include:

1. an uncoupled two state alpha beta filter with position and velocity component states,

2. an uncoupled three state Kalman filter with position, velocity, and acceleration component states, and

3. an extended four Kalman filter with both x and y position and velocity component states.

The first two filters use a one dimensional measurement vector containing the report position component. The extended Kalman filter uses a three dimensional measurement vector containing x and y report position and "pseudo" Doppler, the latter defined as range times range rate.

The tracking software as delivered has been imbedded in a GUI structure that makes it easy to exercise the tracker under a variety of conditions. Using the GUI, the user can interactively select existing or new target scenarios and tracking options prior to tracker execution. Figure 1-1 shows the list of available options.

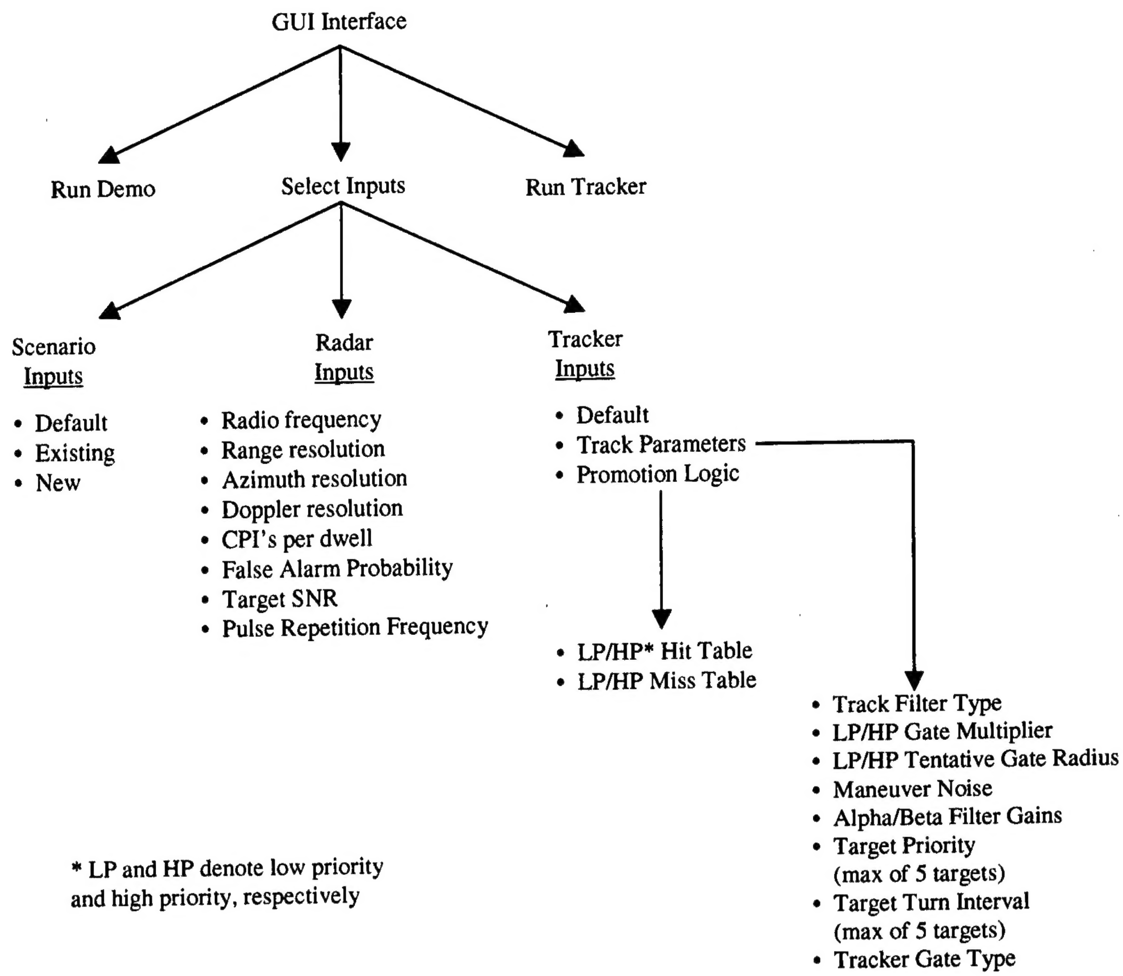

Figure 1-1: Tracker GUI Options

2.0 Tracker Description

The first part of this section contains a narrative description of the basic multiple target tracker. Detailed documentation has been provided in Section 3. This is followed by a discussion of three knowledge based tracking rules that can be used to support the STAP processor. Simple tracking simulations have been provided to illustrate the use of these rules.

2.1 Basic Tracker

A high level description of the multiple target tracker function is shown below in Figure 2-1. The tracker processes reports on a per scan basis and makes use of a track table that contains several track attributes as well as kinematic information. The key attributes include the tracks identification number, its state value, which is a measure of track quality, and its status. Track status can be *dropped, tentative,* or *firm.* In the current software, a dropped status is assigned to a track whose state has been demoted to zero, a tentative status is assigned to a new

track formed by an unused report with a state value initialized to one, and a firm status is given to any track with a state value greater than one. The tracker's function consists of a correlation section together with association and track maintenance sections.

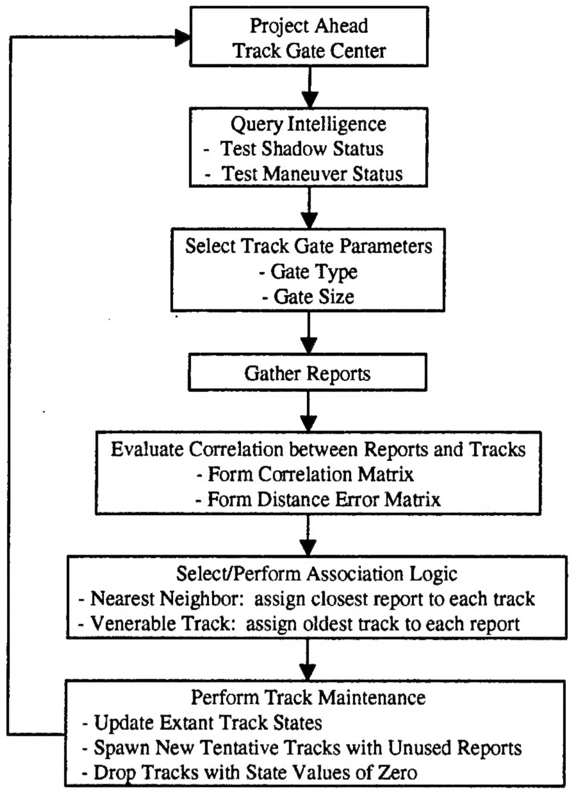

Figure 2-1: High Level Tracker Flow

The correlation section is performed by testing for inclusion each current report against each extrapolated track gate and forming a binary correlation matrix with ones in the capture positions. Tentative track gates formed from unassociated reports are extrapolated by centering a circle of kinematically-determined maximum radium about the report, whereas firm track gates are extrapolated by using the track filter prediction equations. Track gate sizes are typically a function of both the measurement and prediction uncertainties, as well as the track's maneuver status. A distance matrix of the same size as the correlation matrix is also formed containing the report-to-gate center distances.

The association and maintenance section uses the above correlation and distance error arrays to assign reports and tracks in a unique manner. In the even that multiple reports are common to a given track gate, or multiple tracks capture a common report, the association logic will resolve the conflict. Furthermore, any reports that are not assigned to an existing track will be used to spawn a new track designated to have a tentative status. Two simple association logics are currently available. There is a "nearest neighbor" logic that assigns the closest captured report to a given track and there is a "venerable track" logic that associates the oldest common track to a given report.

After all tracks and reports have been associated, the track state promotion logic is applied. Track states are updated using either a "hit" or a "miss" table, depending on whether they correlate with a report. These state tables are usually designed to require specified numbers of successive hits and misses before a track is declared as firm or dropped. Two examples are given below. In Case 1, tables Hit1 and Miss1 allow a tentative track to build up to firm status at state level 4 after three successive this, and to demote with each miss until it reaches state 0 where it is dropped. In Case 2, tables Hit2 and Miss2 show a more complex strategy. A tentative track promotes up to a firm status of 3 after two hits. However, there are hold states 4,5 and 6 which allow the track to recover its firm status more quickly after a single miss.

Case 1				
State	1	2	3	4
Hit1	2	3	4	4
Miss1	0	1	2	3

Case 2						
State	1	2	3	4	5	6
Hit2	2	3	3	5	3	3
Miss2	4	6	5	0	4	0

Once track state logic has been applied, the track table is updated. All tracks with state values of 0 are removed from the table. Furthermore, all reports that were not associated with extant tracks are used to spawn new tentative tracks that are added to the table.

2.2 Knowledge Based Rules

Knowledge based rules make use of extended map and intelligence information, and are used to improve the tracker's ability to support STAP processing requirements. Key issues include the tracking of targets in regions adjacent to large discretes, and in shadow zones that are blocked from the radar-s line of sight. Using map information the tracker should also be able to anticipate a target maneuver that will be required to avoid obstacles. Queued with this information, the tracker will apply appropriately shaped track gates that enhance its capability of capturing reports while maintaining reasonable gate sizes. Four rules are discussed below, along with simple tracking simulations.

In the following examples single target tracks are displayed graphically with the following conventions:

2.2.1 Tracking Legend:

+	Predicted gate center position
*	Measured report position
o	Coasted track position
•	Extrapolated position of a missed detection
d	Dropped tentative track
D#	Dropped firm track
#	Corresponds to age (scans) of dropped track

In addition, all the results were obtained using an uncoupled Kalman tracking filter and all simulations assumed a ten second scan period.

2.2.2 Maneuver/Obstacle Rules:

Both alpha beta and Kalman tracking filters do a good job with targets that move along a constant heading with a fixed speed. Deviations from a straight path cause prediction errors to occur and can ultimately result in a dropped track. Therefore it is important, whenever possible, to anticipate target maneuvers by several scans. This allows time to make such adjustments as increasing the gate size and track gain, or using a shaped gate to allow for across track deviations caused by target turning.

If a track approaches an obstacle whose across track extent is H, a maneuver can be anticipated to occur within a time extent no longer than T_{max}. For this discussion, assuming a constant target speed v and a maximum possible acceleration A_{max}, this extent is given by:

$$T_{\max} = \frac{\rho}{v}\cos^{-1}(1 - H/\rho)$$

where

$$\rho = v^2 / A_{\max}$$

denotes the radius of curvature of the target turn required to clear the obstacle. Somewhere within this time period the tracker should apply its maneuver logic.

Figures 2-2 to 2-4 use the same section of track to illustrate the effect of different gating strategies on the tracker's ability to handle a maneuvering target. Performance is computed for a constant speed (250 meters/sec), high SNR (20 dB at mid range) Swerling 1 target, as it approaches an obstacle (shaded rectangular region) from below and begins to turn after the twenty third scan.

Figure 2-2 shows the response of an uncompensated tracking filter using a standard track gate centered on the predicted gate center, and oriented along and across range with a size dependent on both the measurement and prediction errors. No target acceleration has been

assumed and no maneuver noise has been added. While the straight line section is handled adequately, the initial track begins to lose the target after the turn begins, whereupon it is demoted to a dropped status on the next four scans. The three kilometer circular gates indicate newly spawned tentative tracks that were subsequently updated to firm status.

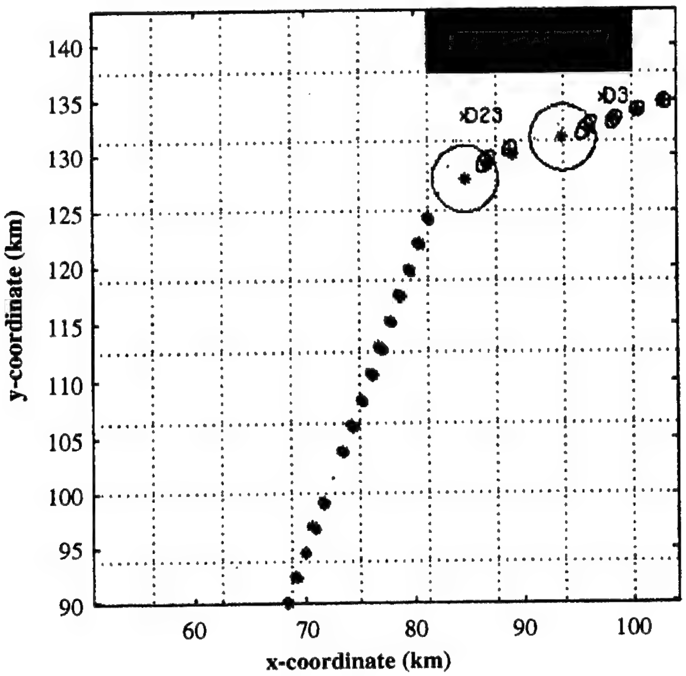

Figure 2-2: Uncompensated Tracker Response for Maneuvering Target

The first maneuvering target rule specifies the use of shaped elliptical gates. Figure 2-3 shows the same target as tracked using a combination of two gates, both centered on the predicted gate center. One gate is oriented along and across range, with a size determined by range and angle measurement uncertainties. The second gate is oriented along and across the targets instantaneous track and its size is a function of the maximum turning acceleration of the target, assumed here to be 0.5g units. Reports capture occurs if the measurement falls within either gate. Except for a missed report early in the linear part of the trajectory, causing the gate to swell initially and then settle down, the track is maintained throughout the maneuver. Each lower case d symbol indicates a dropped tentative track. This occurs when such a track fails to capture a report on the following scan. The large three kilometer circles denote tentative tracks that were successfully promoted to firm status. Finally, the dots occurring in both Figures 2-2 and 2-3, shown extrapolated from the straight line section of the approaching track, represent missed detections. While actual gates existed for these cases, they have been drawn here only or situations in which reports were captured.

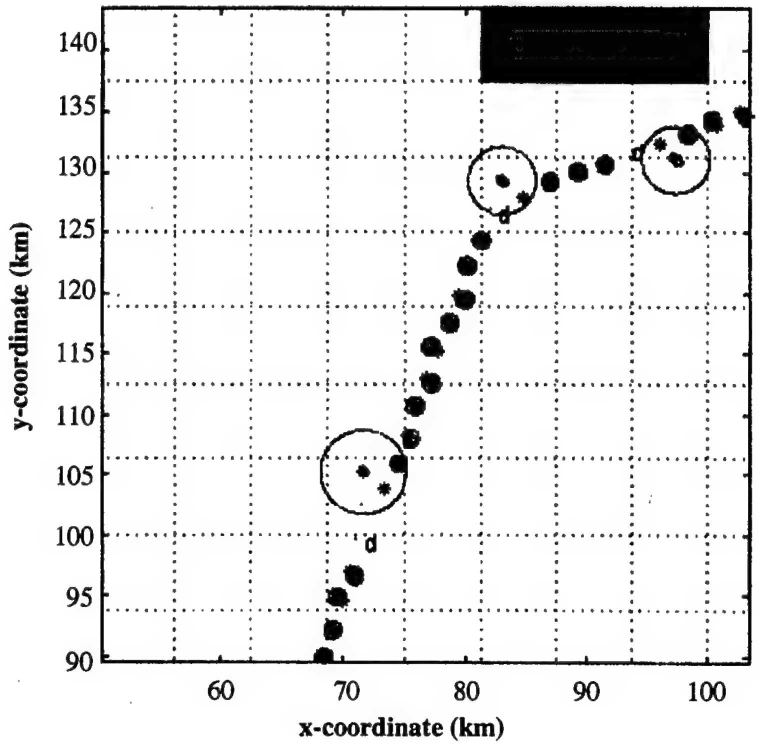

Figure 2-3: Compensated Tracker Response for Maneuvering Target.
Track Oriented Elliptical Gates used with 0.5g Across-Track Acceleration Allowance

The second maneuvering target rule specifies the use of a gate whose shape is determined solely by centripetal turning mechanics. Figure 2-4 shows the target being tracked using a "smile shaped" gate whose shape is determined by the kinematic constraints imposed on a constant speed target undergoing a centripetal maneuver. Let t range over the time interval from the last track update till the next predicted report acquisition, T_{scan} seconds later. The maneuver envelope is given by the xy locus of points, oriented along and across the track, and generated by the equations:

$$x = vt + \rho \sin\theta$$

$$y = \rho(1 - \cos\theta)$$

$$\theta = v(T_{scan} - t) / \rho$$

where the radius of curvature ρ is defined as above. For this gate, a track capture occurs if the measurement ellipse, centered on the measured report and oriented along and across range, intersects the smile locus. As in the previous example, this gate is able to maintain the track throughout the target maneuver. One advantage of this gate is that is has a relatively small area as compared with other maneuver gates and this makes it less likely to capture any false alarms.

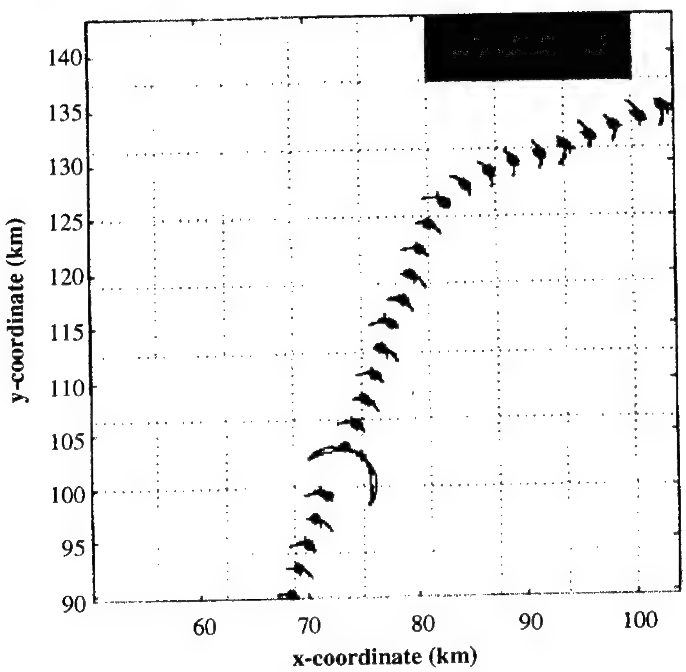

Figure 2-4: Compensated Tracker Response for Maneuvering Target. "Smile" Shaped Centripetal Gates used with (0.0 – 0.5) g Range of Across-Track Acceleration Allowance

2.2.3 Shadow Rule:

The shadow rule provides a means of preserving firm tracks that enter regions shadowed from the radar line of sight. If the predicated track gate center falls within a designed shadow region, both the track state and gate size will be frozen. Upon emerging from the shadow, state promotion resumes and the gate size will not be allowed to exceed a maximum value. In the tracker scenario used in Figures 2-5 and 2-6, only a minimal amount of acceleration noise, 0.05 g units, was used in order to maintain a straight line coasting of the track through the shadow. As previously, the target has a speed of 250 meters per second and the update scan time is 10 seconds.

Figure 2-5 shows a section of tracker response, when no shadow rule is in effect, for a target approaching a shadow zone (shaded rectangle) from below. The three dots denote extrapolated positions of the initial track where no reports were captured. After four demotions the firm track that initially entered the shadow was dropped, as indicated by the D14 symbol. A new tentative track was spawned, and promoted, once the predicted gate positions moved beyond the shadow. Note that the D symbol has been placed at the last updated track position, just prior to entering the shadow, where the track was 14 scans old.

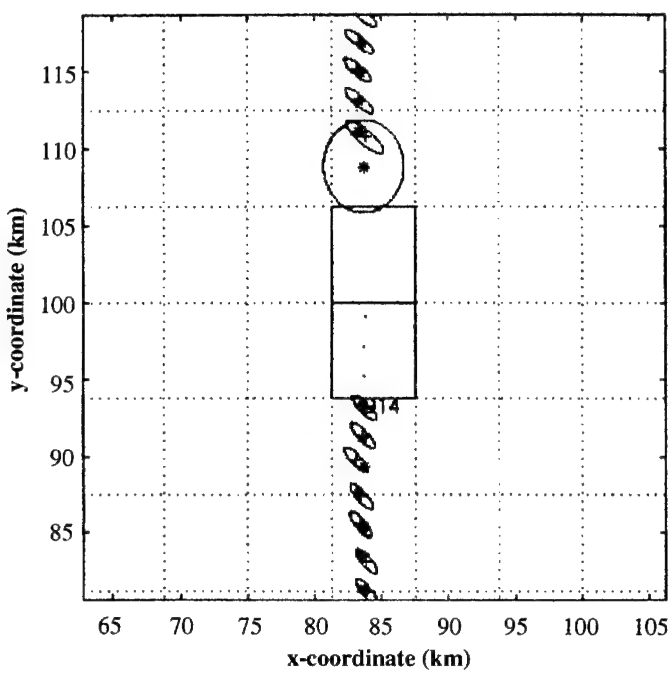

Figure 2-5: Tracker Response for Target Flying through Shadow
No Shadow Rule Applied

Figure 2-6 shows the track history for the same target-shadow scenario when the shadow rule was in place. The circles in the shadow denote coasted track positions at which the track state was held fixed. Once the predicted gate position emerged from the shadow, there was a

moderate increase in gate size, after which it settled down, and the original firm track continued undisturbed.

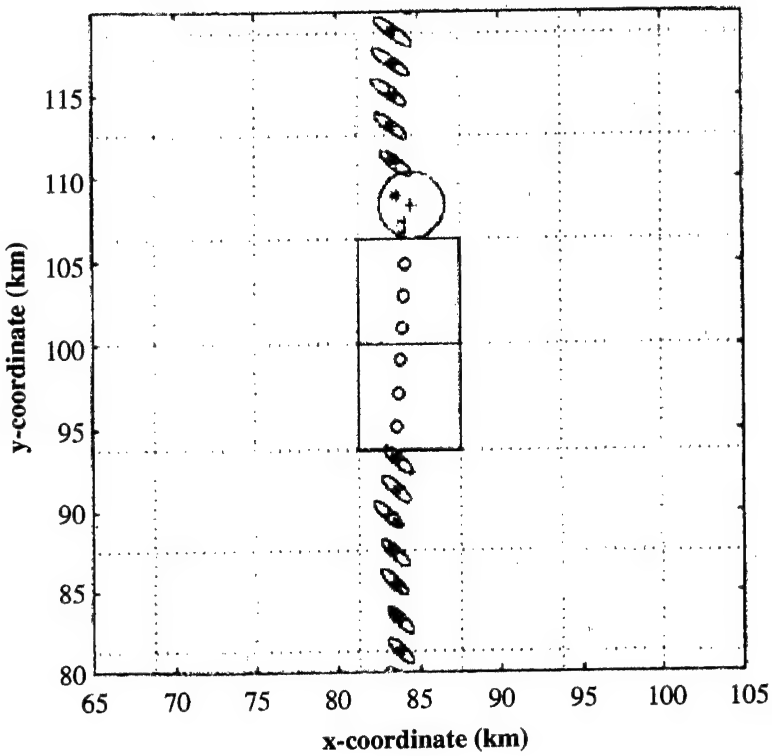

Figure 2-6: Tracker Response for Target Flying through Shadow
Shadow Rule Applied

2.2.4 Discrete Rule:

By tagging large radar returns, or discretes, the STAP processor can exclude regions containing them from its covariance matrix element formation and thereby not use up limited degrees of freedom on their cancellation. The discrete rule allows the tracker to coast through any region containing one of these tagged returns and to essentially ignore it. If a known discrete falls within a track gate, that track will be treated as if in a shadow and will not be updated.

In Figure 2-7, the same target speed, update time, and acceleration noise have been used as in the shadow rule examples. A discrete has been inserted in the target trajectory as shown. As indicated, the tracker preserves the state value of 4 as the predicted gate positions passes through the discrete.

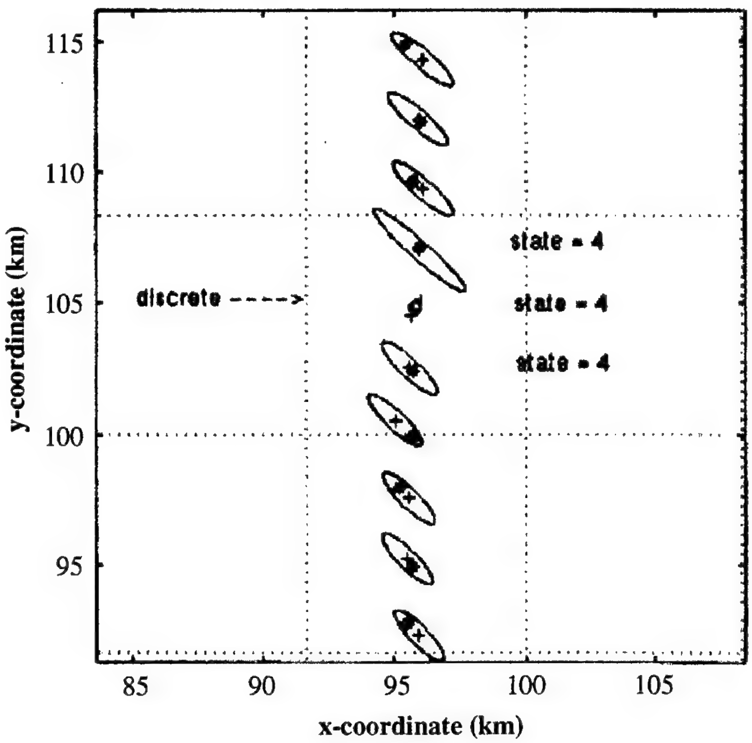

Figure 2-7: Tracker Response for Target Passing Near a Discrete
Discrete Rule Applied

The previous discussion looked at the implementation of four specific tracking rules. However, many more rules that were considered relevant to the STAP problem were developed during the course of this tracking study. In addition to the rule topics discussed here, issues regarding the assignment of target priority, detection threshold, state promotion logic, and other features were considered. A collection of twenty-five such rules are presented in the "Knowledge-Based Tracker Rule Book" that is included in Volume V of this report. Included with each rule is a discussion of its rationale, its impact on the overall knowledge based system and interface requirements that might exist between the tracker, controller and radar.

3.0 Software Description

All tracking software used in this effort has been written in Matlab 5.1. A total of 49 modules are listed and documented below. These can be subdivided into three groupings consisting of user interface, main tracker, and tracker utilities. The first two modules belong to the user interface group. These are Tracker_GUI_7, which provides the graphical user interface for the overall tracking simulation, and Setup_Tracker_6, which sets up the interface between the tracker and the scenario generator. The main tracker group contains the multiple target tracking module, Run_Tracker_8, which carries out all prediction, smoothing, and association operations on report data on a per scan basis. Finally, the tracker utilities group contains all of the support modules that are used by the tracker. It should be noted that some of the modules supporting the user interface group have not been documented since that portion of the software is going to be removed by the customer and replaced with their own hooks into the STAP simulation software.

<u>User Interface Modules</u>

Tracker_GUI_7
Setup_Tracker_6

<u>Main Tracker Module</u>
Run_Tracker_8

<u>Tracker Utility Modules</u>
put_Table_R_2
get_Table_R_2
put_Table_Trk_4
get_Table_Trk_4
add_Table_Trk_4
Cleanup_Table_Trk
print_Table_Trk
Display_Trk_Data_5
State_Update
set_Trackstate
set_TrackHM_stat
set_Track_gates
Shadow_Updates
set_Track_Shadow
predict_shlx_x
Kal_b_pred
Kal_c_pred
smooth_shlx
Kal_b_smth
Kal_c_smth
first_smooth_shlx
abtrack_init
track_init_2

Module Name:	**Tracker_GUI_7**
Calling Module:	None
Called Modules:	All_Defaults_4 Demo_Defaults_4 Read_Scenario_2 Write_Scenario_2 Track_Data_Gen_4 Radar_menu_3 Tracker_Opt_menu_2 Setup_Tracker_6 Run_Tracker_8
Inputs:	None
Outputs:	None
Globals:	***Radar_menu_3*** first_in_Radar f_Ghz N_hits SNR0_dB Prf_kHz DRng_m Az_mrad_3dB Dop_hz Pfa SNR_fa_dB ***Scenario_menu*** script_name Tracker_Opt_resp ***Track_Filter_menu*** Tfilt_resp ***Tracker_param_menu*** first_in_Tr_parm Tfilt_resp Mult_LP Mult_HP radius_TENT_LP radius_TENT_HP Man_amt alpha beta Pri_T Turn_int gate_case ***Tracker_prom_menu*** first_in_Tr_prom Hit_Tbl_LP Miss_Tbl_LP Hit_Tbl_HP Miss_Tbl_HP *** *** fig_no_T Time X-meas Y_meas pDop_meas report_ID X_tru Y_trum pDop_tru S11 S12 S13 S22 S23 S33

ang_pred	Angle (radians) of prediction point
ang_trk	Angle (radians) of track
gate_on	Draw track gate flag (1=> yes)
print_Trk_Tbl	Print track gate flag (1=> yes)
text_on	Label track with text (1=> yes)
N_shadow	Number of shadow zones in scenario
x_sh_LO	Array of low x shadow tile values
x_sh_HI	Array of high x shadow tile values
y_sh_LO	Array of low y shadow tile values
y_sh_HI	Array of high y shadow tile values
In_Shadow	In shadow flag (1=> yes)
Nshadow_dwells	Number of consecutive dwells in shadow

Globals:

Mult
Table_R
Table_T
DROPPED TENT FIRM
LP HP
SUM1_ERROR SUM2_ERROR
P_DET
HP_Buf
dBuf
Display_cnt

Description:

Initializes and sets up variables to be used by the Run_Tracker_8 program. Puts scenario generator outputs in a form usable by tracker. Also adds false alarms to scenario generation data.

Module Name: **Run_Tracker_8**

Calling Modules: None (script file)

Called Modules:

put_Table_R_2
get_Table_R_2
get_Table_Trk_4
put_Table_Trk_4
add_Table_Trk_4
predict_shlx_x
first_smooth_shlx
smooth_shlx
Get_Semiaxes_3
Test_E_Gate
Test_Smile_Gate
get_def_Cov_manu
Assign_Cov_manu
State_Update
set_Trackstate
set_TrackHM_stat
Pred_Shadow_Test
Shadow_Update
Discrete_Test
set_Track_Shadow
get_track_ang
get_pred_ang
update_error
TR_Assoc_Max_T
Display_Trk_Data_5
print_Table_T
Load_dBuf_4
plot_dBuf
Cleanup_Table_Trk

Inputs: setup by Tracker_GUI_7 and Setup_Tracker_6

Outputs: None

Globals: None

Description:

The multiple target tracker processes report data collected during each scan and performs three basic functions: track-report correlation, association, and maintenance.

Module Name: **put_Table_R_2**

Calling Module: Run_Tracker

Called Modules: None

Inputs:

Time	Time of radar blips
X_meas, Y_meas	Measured position (km)
FDop_meas	Measured pseudo Doppler (km*km/sec)
X_tru, Y_tru	True position (km)
FDop_tru	Truer pseudoDoppler (km*km/sec)
S11, …, S33	Measurement Covariance
sig_Rng_km	Range Measurement error (km)
sig_Az_rad	Azimuth measurement error (radians)
sig_Rdot_kmps	Range rate measurement error (km/sec)
freq_GHz	Radio frequency of radar (gHz)
Det_Level	Detection level
Prior_intel	Priority status
Manu_intel	Maneuver status
report_ID	Trajectory ID of report
Nrep	Number of reports calling Module: Run_Tracker

Called Modules: None

Inputs:

Time	Time of radar blips (sec)
X_meas, Y_meas	Measured position (km)
FDop_meas	Measured pseudo Doppler (km*km/sec)
X_tru, Y_tru	True position (km)
FDop_tru	True pseudoDoppler (km*km/sec)
S11, …, S33	Measurement Covariance
sig_Rng_km	Range Measurement error (km)
sig_Az_rad	Azimuth measurement error (radians)
sig_Rdot_kmps	Range rate measurement error (km/sec)
freq_GHz	Radio frequency of radar (gHz)
Det_Level	Detection level
Prior_intel	Priority status
Manu_intel	Maneuver status
report_ID	Trajectory ID of report
Nrep	Number of reports calling Module: Run_Tracker
scan	Current scan index

Outputs: Table_R

Globals: Table_R

Description:

Load the report table buffer for the current scan. All arrays are indexed as (report, scan) and were created by a scenario generator.

Module Name: **get_Table_R_2**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

rep	report index

Outputs:

T_m	Blip time (sec)
X_m, Y_m	Measured position (km)
FDop_m	Measured Doppler (km*km/sec)
Cov_m_vec	Measured covariance vector
sig_rng_km_m	Range measurement error (km)
sig_crng_km_m	Cross range measurement error (km)
sig_Rdot_kmps_m	Range rate measurement error (km?sec)
freq_GHz	Radar rf frequency (gHz)
Det_m	Detection level
priority_in	Priority Status
maneuver_in	Maneuver status
report_source	Trajectory ID
X_tru_m, Y_tru_m	True position (km)
FDop_tru_m	True pseudoDoppler (km*km/sec)
error_status	not used

Globals: Table_R

Description:

Fetch report table data for current scan.

Module Name: **put_Table_Trk_4**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

trk	Track index
TID	Track ID number
status	Track status (drop, tentative, firm)
state	Track state (quality index; 0 => dropped)
report	Captured report index
scan1	Scan index when track became firm
trk_type	Tracker type (alpha beta, unc/extd Kalman)
Z_m	Measurement state vector
Cov_m_vec	Measurement covariance vector
T_m	Measurement of time of captured report
priority	Priority status of track
maneuver	Maneuver status of track
Z_p	Prediction state vector
Cov_p_vec	Prediction covariance vector
Z_s	Smoothed state vector
Cov_s_vec	Smoothed covariance vector
HM_flg	Track capture flag (0 => miss, 1 +. hit)
ang_trk	Track angle (radians)
x_s_2LST	Last track x position (km)
y_s_2LST	Last track y position (km)
T_s_2LST	Last track time (sec)
semi_rng_T	along range semi axis (km)
semi_crng_T	cross range semi axis (km)
semi_trk_T	along track semi axis (km)
semi_ctrk_T	cross track semi axis (km)
In-Shadow	Shadow status flag (0 => not in)
Nshadow_dwells	Number of successive dwells in shadow
gate_case	Track gating choice (0:3)
Accel_max	Maximum acceleration (km/sec*sec)
sig_rng_km	range measurement error (km)
sig_crng_km	cross range measurement error (km)
sig_Rdot_kmps2	range rate meas error (km*km/sec)

Outputs:

error_status
Table_T

Globals: Table_T

Description:

Updates track table data for current scan and report index.

Module Name: **get_Table_Trk_4**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

	trk

Outputs:

	same as inputs to put_Table_Trk_4
TID	same as inputs to put_Table_Trk_4
.	.
.	.
.	.
sig_Rdot_kmps2	.

Globals: Table_T

Description:

Fetch track table data for current scan and report index.

Module Name: **add_Table_Trk_4**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

Ntrk	
num_new_trks	
TID_New	same as inputs to put_Table_Trk4
.	.
.	.
.	.
sig_Rdot_kmps2	.

Outputs:

error_status
Table_T

Globals: Table_T

Description:

Insert new tentative tracks into track table.

Module Name:	**Cleanup_Table_Trk**
Calling Module:	Run_Tracker_8
Called Modules:	None
Inputs:	None
Outputs:	
Ntrk	Number of valid tracks
Globals:	Table_T

Description:

Updates track table for next scan. Sorts Table_T by column 3 (state) and removes all zero states. Counts number of remaining valid tracks and returns this value.

Module Name: **print_Table_T**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

Ntrk	Number of track sets to be printed (max = 10)
sup	vector of Table_T indices to be printed

Outputs: prints to command tool

Globals: Table_T

Description:

Prints track table information as columns (one per track). Support vector sup is subset of {1:86} corresponding to fields in Table_T.

Module Name:	**Display_Trk_Data_5**
Calling Module:	Run_Tracker_8
Called Modules:	Draw_Grid get_Table_Trk_4 get_Table_R_2 Assign_Cov_manu predict_shlx_x get_sig_track get_pred_ang E_Gate_Semiaxes Draw_Smile_Gate Draw_E_Gate_2

Inputs:

Ntrk	Number of tracks to be plotted
scan	current scan index
scan_period	scan time (sec)
sig_rng_km_nom	not used
sig_Az_rad_nom	not used
plot_vec	plot scale: (xmin, xmax, ymin, ymax)
x_v, y_v	x and y grid point sets
dx, dy	x and y grid spacing
gate_on	Draw track gate flag (1 => yes)
text_on	Print Text gate
fig_no	plot figure number
symbol_1	rng-cross rng aligned gate symbol
symbol_2	trk-cross trk aligned gate symbol
semi-max	maximum gate size allowed

Outputs:	graphical display in figure (fig_no)
Globals:	Display_cnt Table_R Table_T DROPPED TENT FIRM LP HP Mult_LP Mult_HP radius_TENT_LP radius_TENT_HP sig_trk_km sig_ctrk_km gate_case Cov_man20 Cov_man30 Cov_man_acc_x Cov_man_acc_y

Description:

The module serves as a diagnostic and display tool for the multi target tracker. It displays predicted and measured positions prior to track table cleanup and draws track gates centered on predicted positions.

Module Name:	**set_Trackstate**
Calling Module:	Run_Tracker_8
Called Modules:	None
Inputs:	

state
trk

Outputs:	None
Globals:	Table_T

Description:

Inputs state value into track table at index trk.

Module Name:	**set_TrackHM_stat**
Calling Module:	Run_Tracker_8
Called Modules:	None
Inputs:	
HM_flg	Capture/shadow status of track
trk	Track index
Outputs:	None
Globals:	Table_T

Description:

Inputs the track capture/shadow status into track table.

Module Name:	**set_Track_gates**
Calling Module:	Run_Tracker_8
Called Modules:	None

Inputs:

semi_rng_T	Along range semi axis of ellipse gate
semi_crng_T	Cross range semi axis
semi_trk_T	Along track semi axis
semi_ctrk_T	Cross track semi axis
trk	Track table index

Outputs:	None
Globals:	Table_T

Description:

Inputs ellipse gate semi axes into track table corresponding to index trk.

Module Name:	**Shadow_Update**
Calling Module:	Run_Tracker_8
Called Modules:	set_Track_Shadow set_TrackHM_stat

Inputs:

trk	Track index into Table_T
In_Shadow	Shadow status flag (1 => in shadow)
Nshadow_dwells_last	Number of scans input trk in shadow
HM_flg_last	Capture/shadow status of input trk

Outputs:	None
Globals:	Table_T

Description:

Updates track shadow status corresponding to index trk. If track is in a shadow, the number of shadow dwells is incremented and the capture status is set to the number 2. Otherwise, the number of dwells is et to zero and the capture status maintained.

Module Name: **set_Track_Shadow**

Calling Module: Run_Tracker_8 Shadow_Update

Called Modules: None

Inputs:

Nshadow_dwells	Number of consecutive dwells in shadow
In_Shadow	in shadow status flag
trk	Index into Table_T

Outputs: None

Globals: Table_T

Description:

Inputs shadow parameters into track table corresponding to index trk. These include (1) the # of consecutive dwells of track in a shadow, and (2) the in-shadow status flag (0 => not in shadow, 1 => in shadow).

Module Name:	**predict_shlx_x**
Calling Module:	Run_Tracker_8
Called Modules:	Kal_b_pred Kal_c_pred

Inputs:

type	Tracking filter type (1:3 +. alpha beta, uncoupled Kalman, and extended Kalman)
Z_s_last	Smoothed state vector at last scan
Cov_s_last_vec	Smoothed covariance array at last scan
Cov_man_x	Maneuver covariance matrix of x component
Cov_man_y	Maneuver covariance matrix of y component
Cov_man	Maneuver covariance matrix of extended Kal
dT	Time interval from last update to present

Outputs:

Z-P	Prediction state vector
Cov_p_vec	Prediction covariance matrix

Globals: None

Description:

Computes predicted state vector and Covariance array, Z_p and Cov_p_vec, respectively. The method used depends on the tracking filter type specified. The maneuver covariance matrices provide a means for inputting acceleration noise to increase track gate size for the Kalman filter cases

Equations:

alpha beta:

X_p = x_s_last + vx_s_last * dT
Y_p = y_s_last + vy_s_last * dT
Vx_p = vx_s_last
Vy_p = vy_s_last

where the components X_p, Vx_p, etc., are related to the state vectors Z_p, Z_s_last, etc. by stacking the x and y components of position, velocity, and acceleration

Z= [X;Vx;Ax;Y;Vy;Ay]

Uncoupled Kalman:

see Kal_b_pred

Extended Kalman:

see Kal_c_pred

Module Name: **Kal_b_pred**

Calling Module: predict_shlx_x

Called Modules: None

Inputs:

Z_s_in	3 x 1	Smoothed input state vector (either component) [Position; Velocity; Acceleration]
dt_in		Time since last update
Cov_manu	3 x 3	Covariance matrix of maneuver noise
Cov_s_in	3 x 3	Smoothed input covariance matrix (either component) [Position; Velocity; Acceleration]

Outputs:

Z_p_out	3 x 1	Prediction state vector (corresponding component)
Cov_p	3 x 3	Prediction covariance matrix

Globals: Table_T

Description:

Computes 3 x 1 prediction state vector and 3 x 3 prediction covariance matrix for the case of an uncoupled Kalman filter. All input and output state vectors are assumed to be 3 x 1, and all covariance matrices are 3 x 3.

Equations:

$$\phi = \begin{bmatrix} 1 & dT & 0.5(dT)^2 \\ 0 & 1 & dT \\ 0 & 0 & 1 \end{bmatrix}; \text{3 x 3 state transition matrix}$$

Z_p_out=ϕ*Z_s_in

Cov_p=ϕ*Cov_s_in*ϕ'+Cov_manu

(ϕ' denotes the transpose of ϕ)

Module Name: **Kal_c_pred**

Calling Modules: predict_shlx_x

Called Modules: None

Inputs:

Z_s_in	4 x 1	Smoothed input state vector [X_s_in; Vx_s_in; Vy_s_in]
dt_in		Time since last update
Cov_manu	4 x 4	Covariance matrix of maneuver noise
Cov_s_in	4 x 4	Smoothed input covariance matrix

Outputs:

Z_p_out	4 x 1	Prediction state vector [X_p_out; Vx_p_out; Y_p_out; Vy_p_out
Cov_p	4 x 4	Prediction covariance matrix

Globals: None

Description:

Computes 4 x 1 prediction state vector and 4 x 4 prediction covariance matrix for the case of an extended Kalman filter. All input and output state vectors are assumed to be 4 x 1, and all covariance matrices are 4 x 4.

Equations:

$$\phi = \begin{bmatrix} 1 & dT & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dT \\ 0 & 0 & 0 & 1 \end{bmatrix}; \text{4 x 4 state transition matrix}$$

$$\text{Z\_p\_out} = \phi * Z\_s\_in$$

$$Cov\_p = \phi * Cov\_s\_in * \phi' + Cov\_man$$

(ϕ' denotes the transpose of ϕ)

Module Name:	**smooth_shlx**
Calling Module:	Run_Tracker_8
Called Modules:	Kal_b_smth Kal_c_smth

Inputs:

type	Tracking filter type (1:3 => alpha beta, uncoupled Kalman, and extended Kalman)
Z_m	Measurement vector (x;y;pDop)
Z_p	Prediction state vector (X;Vx;Ax;Y;Vy;Ay)
Cov_p_vec	Prediction cov array (Cov_p_x;Cov_p_y)reshaped to 1 x 18
Cov_m_vec	Meas cov array 1 x 9
dT	Time interval from last update to present

Outputs:

Z_s	Smoothed state vector (X;Vx;Ax;Y;Vy;Ay)
Cov_s_vec	Smoothed cov array (Cov_s_x;Cov_s_y)reshaped to 1 x 18
Gain_mat	Gain matrix (not used by tracker)

Globals: alpha beta

Description:

Computes smoothed state vector and Covariance array, Z_s and Cov_s_vec, respectively. Also returns a gain matrix (not normally used) which can be used for diagnostic purposes. The smoothing method used depends on the tracking filter type specified.

Equations:

alpha beta:

Dx = x_m – X_p
x_s = X_p + alpha * Dx
Vx_s = Vx_p + beta * Dx/dT

Dy = y_m – Y_p
y_s = Y_p + alpha * Dy
Vy_s = Vy_p + beta * Dy/dT

where

Z_m = [x_m; y_m; pDop_m)

and Z_p and Z_s are 6 x 1 vectors of the position, velocity and acceleration components

Uncoupled Kalman:	see Kal_b_smth
Extended Kalman:	see Kal_c_smth

Module Name: **Kal_b_smth**

Calling Module: smooth_shlx

Called Modules: None

Inputs:

Z_m	3 x 1	Measurement vector [x_m;y_m;pDop_m]
Z_p_out	3 x 1	Prediction state vector (either component) [Position; Velocity; Acceleration]
Cov_p	3 x 3	Covariance matrix of maneuver noise
Cov_s_in	3 x 3	Prediction covariance matrix
var_meas		Measurement variance

Outputs:

Z_s_out	3 x 1	Smoothed output state vector (either component) [Position; Velocity; Acceleration]
Cov_s_out	3 x 3	Smoothed output covariance matrix
Res	3 x 1	Residual error vector (Innovations)
dist_Res_2		Statistical distance
S	3 x 1	Gain matrix

Globals: None

Description:

Computes 3 x 1 smoothed state vector and 3 x 3 smoothed covariance matrix for the case of an uncoupled Kalman filter. All input and output prediction and smoothed state vectors are assumed to be 3 x 1, and all covariance matrices are 3 x 3.

Equations:

(for either x or y component)
M = [1 0 0]; Measurement Matrix (1 Measurement x 3 states)
I_Cov_Res = 1/(M * Cov_p * M' + var_meas)
S = Cov_p * M' * I_Cov_Res; Gain Matrix (3 states x 1 measurement)
Z_s_out = Z_p_out + S * (Z_m – M * Z_p_out)
Cov_s_out = (I – S * M) * Cov_p
Res = (Z_m – M * Z_p) ; Innovations
dist_Res_2 = Res * I_Cov_Res * Res'; Statistical distance

Module Name: **Kal_c_smth**

Calling Modules: smooth_shlx

Called Modules: None

Inputs:

Z_m	3 x 1	Measurement vector (3 x 1) [x_m;y_m;pDop_m]
Z_p_out	4 x 1	Prediction state vector [X; Vx; Y; Vy]
Cov_p	4 x 4	Prediction covariance matrix
Cov_meas	3 x 3	Measurement Covariance

Outputs:

Z_s_out	4 x 1	Smoothed output state vector [X; Vx; Y; Vy]
Cov_s_out	4 x 4	Smoothed output covariance matrix
Res	3 x 1	Residual error vector (Innovations)
dist_Res_2		Statistical distance
S	4 x 3	Gain Matrix

Globals: None

Description:

Computes 4 x 1 smoothed state vector and 4 x 4 smoothed covariance matrix for the case of an extended Kalman filter. All input and output prediction and smoothed state vectors are assumed to be 4 x 1, and all covariance matrices are 4 x 4.

Equations:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ M_{31} & M_{32} & M_{33} & M_{34} \end{bmatrix}$$; Measurement matrix (3 Measurements x 4 states)

where

M_{31} = 0.5 * Z_p_out (2)
M_{32} = 0.5 * Z_p_out (1)
M_{33} = 0.5 * Z_p_out (4)
M_{34} = 0.5 * Z_p_out (3)

Note that

$(pDop)_{est} = M_{31} * X\_p + M_{32} * Vx\_p + M_{33} * Y\_p + M_{34} * Vy\_p$

I_Cov_Res = (M * Cov_p * M' + Cov_meas)-1
S = Cov_p * M' * I_Cov_Res ; 4 x 3 Gain Matrix (4 states x 3 measurements)
Res = (Z_m – M * Z_p_out) ; 3 x 1 Innovations
Z_s_out = Z_p_out + S * Res 4 x 1
Cov_s_out = (I – S * M) * Cov_p 4 x 4
dist_Res_2 = Res; * (I_Cov_Res) * Res

Module Name: **first_smooth_shlx**

Calling Module: Run_Tracker_8

Called Modules:

abtrack_init
track_init_2
track_init_c

Inputs:

type	Track filter type (1:3 => alpha beta, Uncoupled Kalman, and extended Kalman
Z_m_last	Measured vector (last update)
Z_m	Measured vector (current)
Cov_m_vec	Meas Covariance array (current)
dT	Time interval (sec) from last update

Outputs:

Z_s_out	Smoothed state vector
Cov_s_vec	Smoothed covariance array

Globals: None

Description:

Performs smoothing of tentative tracks

Equations:

alpha beta:

see abtrack_init

Uncoupled Kalman: see track_init_2

Extended Kalman: see track_init_c

Module Name:	**abtrack_init**
Calling Module:	first_smooth_shlx
Called Modules:	None

Inputs:

(x1,y1)	Last updated track position
(x2, y2)	Current track position
dT12	Time interval (sec)

Outputs:

(x_s, y_s)	Smoothed track position
(vx_s, vy_s)	Smoothed tracked velocity
ang_trk	Track angle (radians)

Globals: None

Description:

Does a two point initialization of an alpha beta tracker using current and last track positions and their time interval.

Equations:

Dx = (x2 – x1)
x_s = x2
vx_s = Dx/dT12

Dy = (y2 – y1)
y_s = y2
vy_s = Dy/dT12

ang_trk = arctan (Dy/Dx)

Module Name: **track_init_2**

Calling Module: first_smooth_shlx

Called Modules: None

Inputs:

Delt	Time (sec) between current and last update
z_pos_1	Last position component update (x or y)
z_pos_2	Current position component (x or y)
var_pos_m_2	Measurement variance of position component

Outputs:

Z_s_in	Smoothed state vector (3 x 1)
P_s_in	Smoothed covariance matrix (3 x 3)

Globals: None

Description:

Does a two point initialization of an uncoupled Kalman tracker. This routine is applied separately to both the x and y components of target motion.

Equations:

pos_s = z_pos_2
vel_s = (z_pos_2 – z_pos_1)/Delt
Z_s_in = [pos_s; vel_s; 0]

$$P\_s\_in = \begin{bmatrix} Px_{11} & Px_{12} & 0 \\ Px_{12} & Px_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

where

Px_{11} = var_pos_m_2
$Px_{12} = Px_{11}$/Delt
$Px_{22} - Px_{11}$/(Delt)2

Module Name:	**track_init_c**	
Calling Module:	first_smooth_shlx	
Called Modules:	None	

Inputs:

Delt	Time (sec) between current and last update
(x1, y1)	Last position update
(x2, Y2)	Current position
sig_xx	Measurement covariance xx comp
sig_yy	Measurement covariance yy comp
sig_xy	Measurement covariance xy comp

Outputs:

Z_s_in	4 x 1	Smoothed state vector
P_s_in	4 x 4	Smoothed covariance matrix

Globals: None

Description:

Does a two point initialization to be used with the extended Kalman track filter.

Equations:

Let T = Delt, T2 = T^2

$$sig\_x = \sqrt{sig\_xx}, sig\_y = \sqrt{sig\_yy}$$

pos_sx = x2, pos_sy = y2; smoothed positions
vel_sx = (x2 – x1)/T, vel_sy = (y2 – y1)/T ; smoothed velocities
Z_s_in = [pos_sx; vel_sx; pos_sy; vel_sy]

$$P\_s\_in = \begin{matrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{12} & P_{22} & P_{23} & P_{24} \\ P_{13} & P_{23} & P_{33} & P_{34} \\ P_{14} & P_{24} & P_{34} & P_{44} \end{matrix}$$

where

P_{11} = sig_xx, $P_{12} = 1.5 * P_{11}/T$, P_{13} = sig_xy; P_{14} = 1.5 * sig_x * sig_y/T
$P_{22} = 6.5\ P_{11}/T2$; $P_{23} = P_{14}$, P_{24} = sig_xy/T2
P_{33} = sig_yy, $P_{34} = 1.5 * P_{33}/T$
$P_{44} = 6.5 * P_{33}/T2$

Module Name: **Assign_Cov_manu**

Calling Module: Run_Tracker_8

Called Modules: get_def_Cov_manu

Inputs:

trk_type	Track filter type
scan	Current scan
manu_rep	Maneuver anticipation status flag
rep_source	Maneuver anticipation interval (scan lo, scan hi)

Outputs:

Cov_man	Maneuver covariance matrix
Cov_man_x	
Cov_man_y	

Globals:

Cov_man20
Cov_man30
Cov_man_acc_x
Cov_man_acc_y
Turn_int

Description:

Sets up the maneuver covariance matrix. If a maneuver has been anticipated for the current scan then maneuver noise is applied. Otherwise a small value (essentially zero) is loaded into the matrix.

Equations:

Accel_max = Maximum maneuver acceleration (km/s^2)
var_acc = $(\text{Accel\_max})^2$
var_vel = $(\text{Accel\_max} * \text{scan\_period})^2$
var_pos = $(0.5 * \text{Accel\_max} * (\text{scan\_period})^2)^2$

$$\text{Cov\_man} = \begin{bmatrix} \mathbf{var\_}pos & 0 & 0 \\ 0 & \mathbf{var\_}vel & 0 \\ 0 & 0 & \mathbf{var\_}acc \end{bmatrix}; \text{trk\_type} \neq 3$$

$$\text{Cov\_man} = \begin{bmatrix} \mathbf{var\_}pos & 0 & 0 & 0 \\ 0 & \mathbf{var\_}vel & 0 & 0 \\ 0 & 0 & \mathbf{var\_}pos & 0 \\ 0 & 0 & 0 & \mathbf{var\_}vel \end{bmatrix}; \text{trk\_type} = 3$$

Cov_man_x = Cov_man_y = Cov_man

Module Name: **get_def_Cov_manu**

Calling Module: Assign_Cov_manu

Called Modules: None

Inputs:

trk_type Track filter type

Outputs:

Cov_man Maneuver covariance matrix
Cov_man_x
Cov_man_y

Globals:

Cov_man20
Cov_man30

Description:

Loads default values into maneuver covariance matrices.

Equations:

Cov_man = Cov_man20; $\quad$ trk_type $\neq 3$
Cov_man = Cov_man30; $\quad$ trk_type $= 3$
Cov_man_x = Cov_man_y = Cov_man;

where

Cov_man20 = le-10 * eye (3)
Cov_man30 = le-6 * eye (4)

Module Name: **Get_Semiaxes_3**

Calling Module: Run_Tracker_8

Called Modules:

E_Gate_Semiaxes_3
get_sig_track
Scale_Shadow_gates

Inputs:

In_Shadow	Current in shadow flag (1 => yes)
In_Shadow_last	Last update in shadow flag
Nshadow_dwells_last	Number of successive dwells in shadow as of last update
gate_case	Gate type flag (0:3)
trk_type	Track filter type (1:3)
Mult	Gate size multiplier
Accel_max	Maximum assumed acceleration
dT	Time (sec) since last update
ang_pred	Angle (rad) of predicted point wrt radar cs
ang_trk	Angle (rad) of track wrt radar cs
sig_rng_km_m	Range measurement error (km)
sig_crng_km_m	Cross range measurement error (km)
Z_s_last	Smoothed state vector (last update)
Z_p	Prediction state vector (current)
Cov_p_vec	prediction covariance array (current)
semi-rng_T_last	Along range semi axes (last)
semi_crng_T_last	Cross range semi axes (last)
semi_trk_T_last	Along track semi axes (last)
semi_ctrk_T_last	Cross track semi axes (last)
semi_max	Maximum allowed semi axis size

Outputs:

semi_rng_T	Along range semi axes (current)
semi_crng_T	Cross range semi axes (current)
semi_trk_T	Along track semi axes (current)
semi_ctrk_T	Cross track semi axes (current)

Globals: None

Description:

Computes the semi axes for the two orientations of elliptical gates used. The first type is oriented along and across range and the second is oriented along and across track. Tracks that are in a shadow have their gate sizes frozen. Otherwise, track gates sizes are determined as a function of the gate case and track filter type selected. This function is done by the routine E_Gate_Semiaxes_3.

Module Name: **E_Gate_Semiaxes_3**

Calling Module: Get_Semiaxes_3

Called Modules: Rotate_xy2xpyp

Inputs:

gate_case	Gate type flag (0:3)
trk_type	Track filter type (1:3)
Mult	Gate size multiplier
Z_p	Prediction state vector (current)
Cov_p_vec	prediction covariance array (current)
ang_pred	Angle (rad) of predicted point wrt radar cs
sig_rng	Range measurement error (km)
sig_crng	Cross range measurement error (km)
ang_trk	Angle (rad) of track wrt radar cs
sig_trk	Along track measurement error (km)
sig_ctrk	Cross track measurement error (km)
semi_max	Maximum allowed semi axes size

Outputs:

semi_rng	Along range semi axes (km)
semi_crng	Cross range semi axes (km)
semi_trk	Along track semi axes (km)
semi_ctrk	Cross track semi axes (km)

Globals: None

Description:

Computes semi axes for elliptical gates oriented along/cross range, and oriented along/cross track. Results depend on which of three gate cases are chosen (cases 1 and 2 require a Kalman filter). In each case the along/cross track gate sizes are computed as scaled versions of the along/cross track measurement errors. However, the along/cross range oriented gate sizes are case dependent. Gate case 1 combines in an rss fashion (1) range and cross range measurement errors from the last track update, and, (2) current prediction covariance estimates in x and y, projected onto the range/cross range axes. The composite gate is formed by multiplying this result by a scale factor. Gate case 2 uses only the scaled prediction covariance estimates in x and y to form the gate. Gate case 3 uses the scaled along/cross range measurement errors to form this gate.

Equations:

The quantities sig_rng and sig_crng are measurement errors along range and cross range computed from the last report captured by a given track and stored in the track table. The quantities sig_trk and sig_ctrk are based on kinematical assumptions and are computed in get_sig_track.

alpha beta filter:

$$\begin{bmatrix} \text{semi\_rng} \\ \text{semi\_crng} \\ \text{semi\_trk} \\ \text{semi\_ctrk} \end{bmatrix} = \begin{bmatrix} \text{sig\_rng} \\ \text{sig\_crng} \\ \text{sig\_trk} \\ \text{sig\_ctrk} \end{bmatrix} * \begin{cases} 1.5 Mult, (\text{gate\_case} < 3) \\ Mult \quad , (\text{gate\_case} = 3) \end{cases}$$

Kalman filters:

Get prediction uncertainty along x and y directions (σ_{px}, σ_{py}) from the covariance array Cov_p_vec.

(gate_case = 1)

$\Delta\theta$ = (ang_pred – ang_trk)
Compute projections of σ_{px} and σ_{py} onto the range-cross range axes.

$$\begin{bmatrix} \sigma_{px'} \\ \sigma_{py'} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) \\ -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \sigma_{px} \\ \sigma_{py} \end{bmatrix}$$

Form the "root sum square" (Rss) of $\sigma_{px'}$ and $\sigma_{py'}$ with the measurement errors

$$\text{semi\_rng} = \text{Mult} * \sqrt{(\sigma px')^2 + (\text{sig\_rng})^2}$$

$$\text{semi\_crng} = \text{Mult} * \sqrt{(\sigma px')^2 + (\text{sig\_crng})^2}$$

semi_trk = Mult * sig_trk

semi_ctrk = Mult * sig_ctrk

(gate_case = 2)

θ = ang_trk

Compute projections of σ_{px} and σ_{py} onto track-cross track axes.

$$\begin{bmatrix} \sigma_{px''} \\ \sigma_{y''} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \sigma_{px} \\ \sigma_{py} \end{bmatrix}$$

$$\begin{bmatrix} \text{semi\_rng} \\ \text{semi\_crng} \\ \text{semi\_trk} \\ \text{semi\_ctrk} \end{bmatrix} = \text{Mult} * \begin{bmatrix} \text{sig\_rng} \\ \text{sig\_crng} \\ \left|\sigma_{px''}\right| \\ \left|\sigma_{py''}\right| \end{bmatrix}$$

(gate_case = 3)

$$\begin{bmatrix} \text{semi\_rng} \\ \text{semi\_crng} \\ \text{semi\_trk} \\ \text{semi\_ctrk} \end{bmatrix} = \text{Mult} * \begin{bmatrix} \text{sig\_rng} \\ \text{sig\_crng} \\ \text{sig\_trk} \\ \text{sig\_ctrk} \end{bmatrix}$$

Module Name: **get_sig_track**

Calling Module: Get_Semiaxes_3

Called Modules: None

Inputs:

Z_s	Smoothed state vector
Accel_max	Maximum allowed acceleration (km/s*s)
dT	Time (sec) since last update

Outputs:

sig_trk_km	Along track uncertainty (km)
sig_ctrk_km	Cross track uncertainty (km)

Globals: None

Description:

Computes along/cross track uncertainties. The along track uncertainty is computed as $0.5*A*(dT)^2$ with the along track acceleration A assumed bounded by 0.5g. The across track uncertainty is assumed to be due to constant speed turning only.

Equations:

Define the following quantities:

Acc_at = along track acceleration ≤ Accel_max
V_kmps = estimated track speed
ρ_{km} = radius of curvature of turn (km)
Ang = angle of turn voer time dT (radians)

$\rho_{km} = (V\_kmps)^2/Accel\_max$
$ang = V\_kmps * dT/\rho_{km}$

$sig\_trk\_km = 0.5\ Acc\_at\ (dT)^2$
$sig\_ctrk\_km = \rho_{km}\ (1 - \cos(Ang))$

Module Name:	**Scale_Shadow_Gates**
Calling Module:	Get_Semiaxes_3
Called Modules:	None

Inputs:

gate_scale	Gate size multiplier
semi-rng_T_in	Along range semi axis (input)
semi_crng_T_in	Cross range semi axis (input)
semi_trk_T_in	Along track semi axis (input)
semi_ctrk_T_in	Cross track semi axis (input)

Outputs:

semi_rng_T_out	Along range semi axis (output)
semi_crng_T_out	Cross range semi axis (output)
semi_trk_T_out	Along track semi axis (output)
semi_ctrk_T_out	Cross track semi axis (output)

Globals: None

Description:

Multiplies input gate semi axes by a scale factor.

Module Name:	**Rotate_xy2xpyp**
Calling Module:	E_Gate_Semiaxes_3
Called Modules:	None

Inputs:

ang_rad	Rotation angle (rad)
Z	Input two component vector

Outputs:	Rotated two component vector
Globals:	None

Description:

Rotates the two component vector Z through angle ang_rad.

Equations:

θ = ang_rad and

Let $$\text{Rot} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

Z_p = Rot * Z

Module Name: **Test_E_Gate**

Calling Module: Run_Tracker_8

Called Modules: Rotate_xy2xpyp

Inputs:

Z_m	Measurement state vector (x;y;pDop)
Z_p	Prediction state vector (x;vx;ax;y;vy;ay)
ang_rotn	Rotation angle wrt radar cs (radians)
semi_along	Semi axis length along rotated x
semi_across	Semi axis length along rotated y

Outputs:

In_E_Gate	Inclusion test flag (1 => inclusion)

Globals: None

Description:

Tests if measured point Z_m lies within an ellipse which is (1) oriented at an angle "ang_rotn" with respect to the radar coordinate system, (2) centered on the predicted position, Z_p, and, (3) has semi axes of length semi_along and semi_across. Returns a value of zero if test fails, and a value of one if test passes.

Equations:

Let (X_m, Y_m) and (X_p, Y_p) denote the measured predicted points, respectively. Transform the predicted-measurement position error into the local coordinate system of the ellipse, centered on the predicted point. The transformed residual errors $(\Delta X', \Delta Y')$ are

$$\begin{bmatrix} \Delta X' \\ \Delta Y' \end{bmatrix} = \text{Rot} * \begin{bmatrix} X_p - X_m \\ Y_p - Y_m \end{bmatrix}$$

where

$$Y_e = \text{semi\_across} * \sqrt{1 - (\Delta X' / \text{semi\_along})^2}$$

The condition for inclusion of (X_m, Y_m) within the ellipse is

$$|\Delta Y'| \leq Y_e \quad \text{and}$$
$$|\Delta X'| \leq \text{semi\_along}$$

Module Name: **Test_Smile_Gate**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

T_s	Scan period (sec)
Z_m	Measurement state vector (x;y;pDop)
Z_p	Prediction state vector (x;vx;ax;y;vy;ay)
Z_s_last	Smoothed state vector (last update)
x_last_km	x position (last update)
y_last_km	y position (last update)
ag_HI	Maximum turn acceleration (g units)
semi_rng	Measurement ellipse semi axis along range
semi_crng	Measurement ellipse semi axis cross range

Outputs:

In_gate_m	Inclusion test flag (1 => included)

Globals: None

Description:

Tests if measurement ellipse intersects centripetal maneuver "smile" shaped gate.

Equations:

The smile gate envelope is defined by two equations that are parameterized by the time t, ranging from the current report time to one scan period later. Let v denote the constant target speed and ρ be the radius of curvature of the turn. Then

$\rho = v^2/(9.8 * ag\_HI)$ and
$Ang = v\,(T\_s - t)/\rho$.

With these the equations for the maneuver envelope in the local track coordinate system, centered on the last smoothed position, are

$$x_s' = vt + \rho\,\mathbf{sin}(ang)$$
$$y_s' = \rho(1 - \mathbf{cos}(ang))$$

The gate test is performed by:

(1) Transforming the boundary points (x'_s, y'_s) to the radar coordinate system

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x\_last\_km \\ y\_last\_km \end{bmatrix} + \begin{bmatrix} \cos(\theta_{trk}) & -\sin(\theta_{trk}) \\ \sin(\theta_{trk}) & \cos(\theta_{trk}) \end{bmatrix} \begin{bmatrix} x'_s \\ y'_s \end{bmatrix}$$

where θ_{trk} denotes the instantaneous track angle.

(2) Transforming (x_s, y_s) to local coordinate system centered on the measured point and oriented at the angle of the measured point θ_m.

$$\begin{bmatrix} \Delta X''_s \\ \Delta Y''_s \end{bmatrix} = \begin{bmatrix} \cos\theta_m & \sin\theta_m \\ -\sin\theta_m & \cos(\theta_m) \end{bmatrix} \begin{bmatrix} x_s - x_m \\ y_s - y_m \end{bmatrix}$$

(3) Test each maneuver envelope point for inclusion within the measurement ellipse

$$|\Delta X''_s| \leq semi\_rng$$
$$|\Delta Y''_s| \leq Y_e$$
$$Y_e = semi\_crng \sqrt{1 - (\Delta X''_s / semi\_rng)^2}$$

Module Name:	**Pred_Shadow_Test**
Calling Module:	Run_Tracker_8
Called Modules:	
	get_def_Cov_manu predict_shlx_x Test_In_Shadow

Inputs:

trk_type	Type of tracking filter (1:3)
Z_s_last	Smoothed state vector (last update)
Cov_s_last_vec	Smoothed covariance array (last update)
dT	Time interval (sec) since last update

Outputs:

Z_p	Prediction state vector
In_Shadow	Shadow status flag (1 => in shadow)

Globals:

Cov_man20 Cov_man30
x_sh_LO x_sh_HI y_sh_LO y_sh_HI

Description:

Tests if predicted position falls within a shadow region. Sets flag to 1 if in shadow. [The shadow zones were specified as a set of rectangular tiles by the scenario generator. The array x_sh_LO, x_sh_HI, y_sh_LO, and, y_sh_HI specify low and high positions of each time.]

Module Name:	**Test_In_Shadow**
Calling Module:	Shadow_Test
Called Modules:	None

Inputs:

Z_p	Prediction state vector (x;vx;ax;y;vy;ay)
x_sh_LO	Array specifying low x position of shadow tiles
x_sh_HI	Array specifying high x position of shadow tiles
y_sh_LO	Array specifying low y position of shadow tiles
y_sh_HI	Array specifying high y position of shadow tiles

Outputs:

In_Shadow	In shadow status flag (1 => in shadow)

Globals:	None

Description:

Tests if predicted position falls within any of the shadow tiles that were specified by the scenario generator.

Module Name: **Discrete_Test**

Calling Module: Run_Tracker_8

Called Modules: E_Discrete_Test

Inputs:

Z_p	Prediction state vector
gate_case	Gate case flag (1:3)
ang_pred	Angle (rad) of predicted point
semi_rng_T	Semi axis length of along range ellipse
semi_crng_T	Semi axis length of cross range ellipse
ang_trk	Angle (rad) of track
semi_trk_T	Semi axis length of along track ellipse
semi_ctrk_T	Semi axis length of cross track ellipse

Outputs:

In_Discrete	Discrete capture flag (1 => discrete present)

Globals: None

Description:

Tests if discrete point falls within either of two gates oriented along/across range or along/cross track.

Equations:

(gate_case = 1)

Each discrete point is tested for inclusion in a single ellipse, centered on the predicted point, and oriented along range/cross range. The semi axes are given by semi_rng_T and semi_cross_T.

(gate_case >1)

Each discrete point is tested for inclusion if either of two ellipses, both centered on the predicted point. The first ellipse is the same as defined above. The second ellipse is oriented along track/cross track and has semi axes given by semi_trk_T and semi_ctrk_T.

Module Name:	**E_Discrete_Test**
Calling Module:	Discrete_test
Called Modules:	Test_E_Gate

Inputs:

Z_p	Prediction state vector (x;vx;ax;y;vy;ay)
ang_rotn	Rotation angle (rad) of ellipse gate wrt radar cs
semi_along	Semi axis length of gate along rotated x
semi_across	Semi axis length of gate along rotated y

Outputs:

In_Discrete	Discrete inclusion flag (1 => discrete in)

Globals: Ndiscrete X_D_Vec Y_D_Vec

Description:

Tests if any of a set of discrete points falls within an elliptical gate: (1) centered on the predicted point, (2) oriented at an angle "ang_rotn" wrt radar coordination system, (3) having semi axes lengths of semi_along along the rotated x axis and of length semi_across along the rotated y axis.

The discrete locations were specified by the scenario generator and given here by the arrays X_D_Vec and Y_D_Vec.

Module Name:	**TR_Assoc_Max_T**
Calling Module:	Run_Tracker_8
Called Modules:	cmp_track_age

Inputs:

Corr	Binary correlation matrix (trks, reps)
Dist	Distance matrix (trks, reps)
NAT_in	Number of active tracks in Table_T
Nrep_in	Number of reports from current scan
scan	Current scan

Outputs:

Update	Binary array of updated tracks (1 => updated)
unAssoc	Binary array of unassociated tracks
unused	Binary array of unused reports
Asgn_Reps	Array of reports assigned to each track

Globals:	Table_T

Description:

Assigns unique reports to corresponding track. If a report is common to multiple tracks then it is assigned to the oldest track.

Module Name:	**TR_Assoc_Min_D**
Calling Module:	Run_Tracker_8
Called Modules:	cmp_track_age

Inputs:

Corr	Binary correlation matrix (trks, reps)
Dist	Distance matrix (trks, reps)
NAT_in	Number of active tracks in Table_T
Nrep_in	Number of reports from current scan
scan	Current scan

Outputs:

Update	Binary array of updated tracks (1 => updated)
unAssoc	Binary array of unassociated tracks
unused	Binary array of unused reports
Asgn_Reps	Array of reports assigned to each track

Globals:	Table_T

Description:

Assigns unique reports to corresponding track. If multiple reports are common to a given track then the closest is assigned to the track.

Module Name:	**cmp_track_age**
Calling Module:	TR_Assoc_Max_T
Called Modules:	None
Inputs:	
trk	Track index into Table_T
scan	Current scan
Outputs:	
age	Track age since first became firm
Globals:	Table_T

Description:

Computes age of a track in scan units. Age is defined as time since track became firm.

Module Name:	**get_track_ang**
Calling Module:	Run_Tracker_8
Called Modules:	None
Inputs:	
Z_p	Prediction state vector
Z_s_last	Smoothed state vector (last update)
Outputs:	
ang_trk	Angle of track (radians)
Globals:	None

Description:

Computes track angle in radians with respect to the radar coordinate system.

Module Name: **get_pred_ang**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

Z_p Prediction state vector

Outputs:

ang_pred Prediction of Angle

Globals: None

Description:

Computes angle of predicted point with respect to the radar coordinate system.

Module Name: **update_error**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

N_Hits	Number of times track captured a report
Z_tru	True state vector
Z_p	Prediction state vector
TID	Track ID number

Outputs:

mean_error	Position error (predicted – true) averaged over N_Hits
sigma_error	Standard deviation of position error

Globals:

SUM1_ERROR
SUM2_ERROR

Description:

Computes running mean and standard deviation of position error between predicted and true values as a function of track ID number. Stores running first and second moments in global buffers SUM1_ERROR and SUM2_ERROR.

Module Name: **Load_P_DET**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

Ntk_max	Number of tracks in P_DET buffer
scan	Current scan index
N_HITS	Number of captures for track
N_EVENTS	Number of captures plus misses for track

Outputs: None

Globals: Table_T_P_DET

Description:

Computes fraction of captures for specific track over its evolution. Stores results in global buffer P_DET.

Module Name: **Load_dBuf_4**

Calling Module: Run_Tracker_8

Called Modules: get_Table_Trk_4

Inputs:

Ntrk	Current number of tracks
dBuf_last_cnt	Number of dropped HP tracks in buffer dBuf at last scan

Outputs:

dBuf_cnt	Number of dropped HP tracks in buffer dBuf currently

Globals:

dBuf
Table_T
DROPPED TENT FIRM
DP LP

Description:

Updates the high priority dropped track buffer each scan. The buffer dBuf has three components: (time, x position, y position)

Module Name: **plot_dBuf**

Calling Module: Run_Tracker_8

Called Modules: None

Inputs:

dBuf_new	Counter in dropped track buffer
plot_vec	Min and max s and y values of plot space

Outputs: None

Globals: dBuf

Description:

Plots dropped high priority tracks each scan.

☆U.S. GOVERNMENT PRINTING OFFICE: 2001-610-055-10150